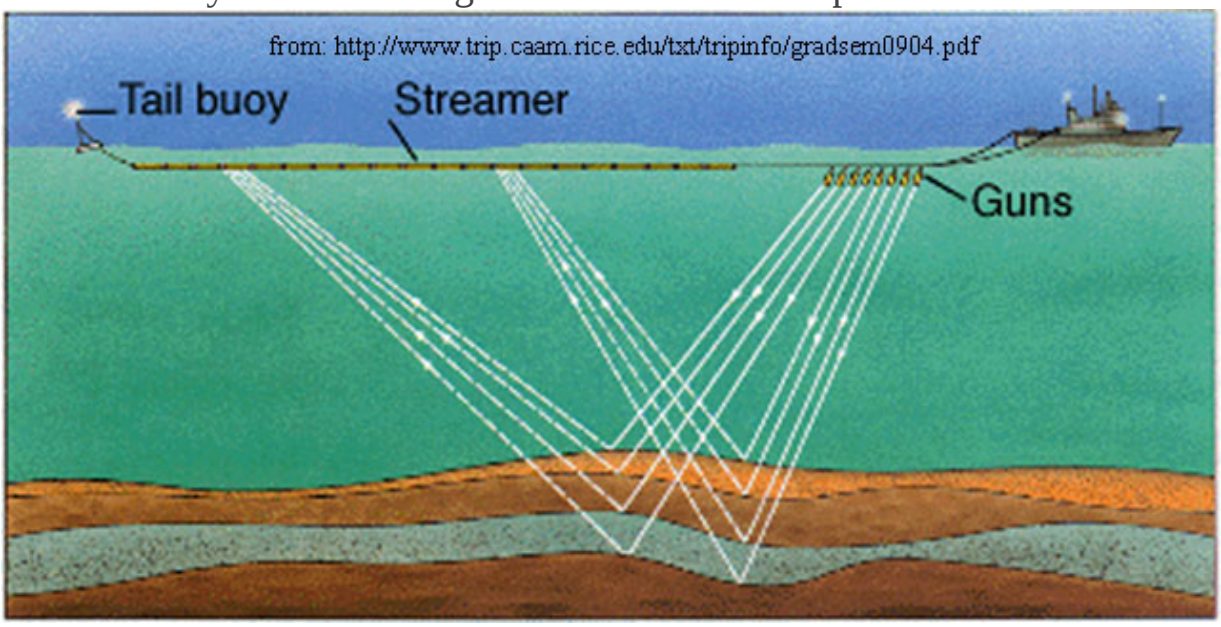
Introduction to Seismic Imaging

Living in a time where natural resources are scarce and precious, it is important to find accurate ways of mapping surfaces such as underwater landmarks or the Earth's interior. A considerable amount of money and effort has been spent on the field of reflection seismology, the science of collecting echoes and transforming them into images of surfaces.

One method to accomplish such a task is multi-offset Kirchoff migration. This technique employs various sources and receivers placed apart from each other. Each source fires an acoustical pulse that reflects off of the surface to be mapped. The echo is then collected at each receiver.

Our goal was to construct a digital image of a reflecting surface based on the time delays between the generated and received pulses.



Artist's rendering of an imaging experiment.

Mathematical Foundations

Sound waves travel at definite velocities characteristic of the propagation medium. Reflection occurs when the waves hit a change in medium and experience a change in velocity. Fermat's principle of least time states that a wave, in going between two points S and R, must traverse a path length that is stationary with respect to variations of that path. While this principle was formulated for light beams, it also holds for other waves.

Fermat's principle can be understood from a phasor perspective. Waves that traverse paths close to that of a maximum or minimum (stationary) path will arrive by routes that only differ slightly in path length. Hence, they will arrive from S to R nearly in-phase, and they will add constructively. Waves taking other paths far away from the stationary one will arrive mainly out of phase with each other and cancel.

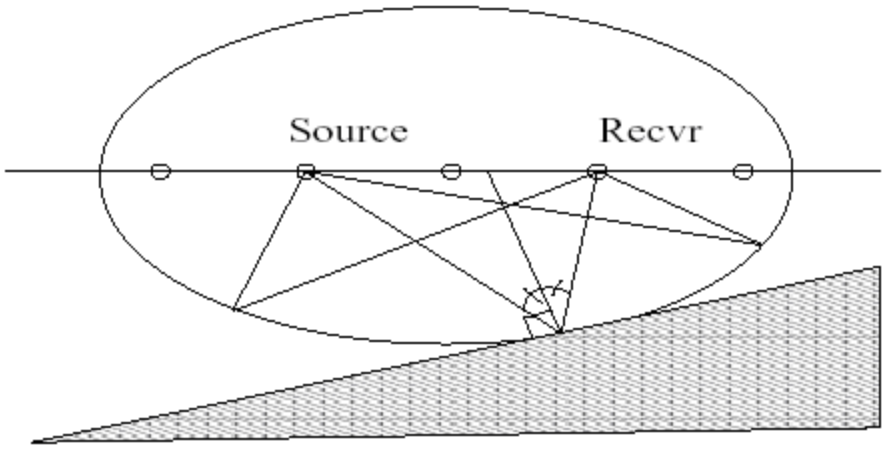
For an ellipse, the sum of the distances from the two focal points to a point on the ellipse is a constant. Thus the paths SQR for all points Q on an ellipse are stationary. If a pulse is fired from the source S and arrives at receiver R a time t later, we can then place the source and receive at the focal points of an ellipse. The sum of the distances from each focal point to a point on the ellipse is given by:

Equation:

$$r_1 + r_2 = vt$$

where v is the speed of sound in the given medium. The ellipse represents the locus of all possible image points for one source/receiver pair.

Source and Receiver diagram demonstrating the locus of possible reflection points.



This way, we can draw such an ellipse for each source-receiver pair. Each image exhibits elliptical wavefronts. The intensity of an oscillatory integral is given by:

Equation:

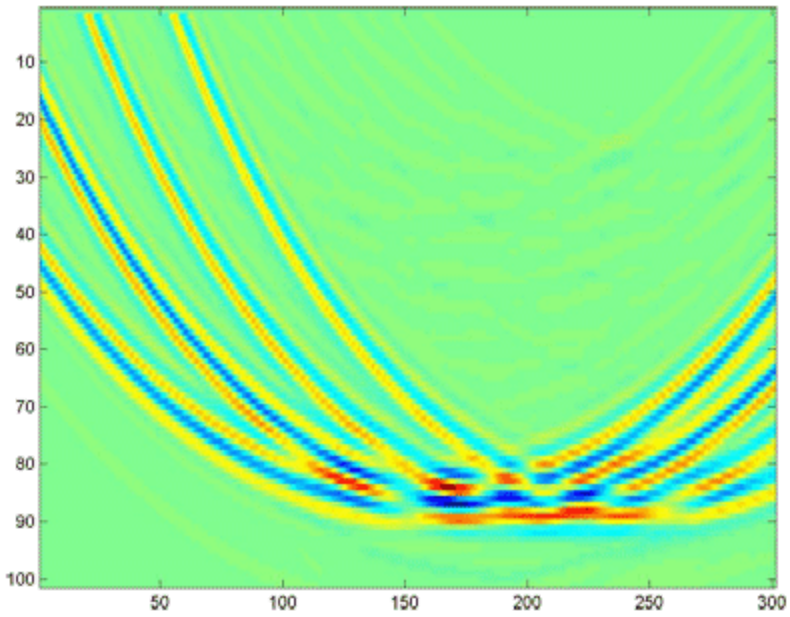
$$I(\omega) = \quad \mathrm{d}\, x a(x) e^{iw\psi(x)}$$

where a(x) is the amplitude function, ω is the frequency, and φ(x) is the phase of the signal. Because we are sending a pulse, we can make a large ω approximation and look at the case where . For a small section dx, the high number of oscillations due to large ω causes the integral to equal zero except where

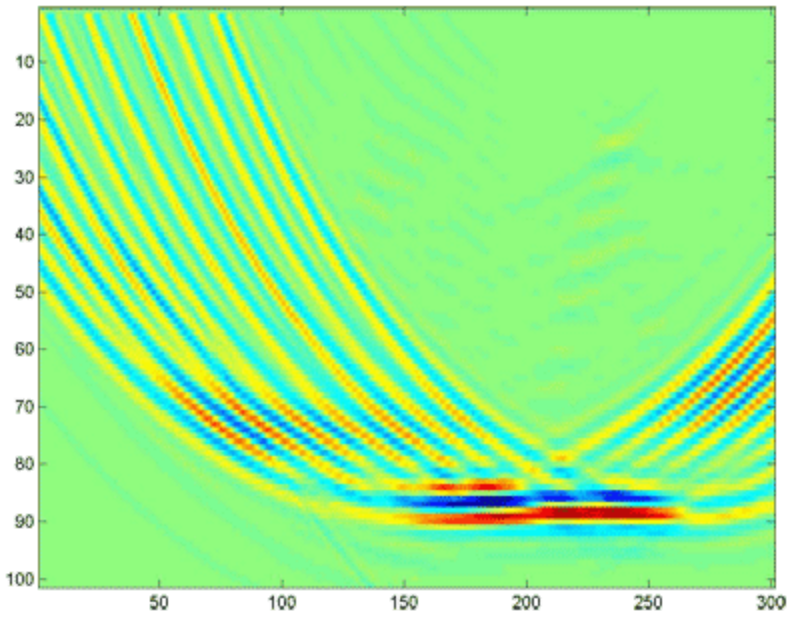
$$\frac{d\phi}{dx} = 0$$

. This point is a point of stationary phase, and occurs where the wavefront is. Therefore, after summing up all the ellipses generated by each source-receiver pair, only the outline of the reflecting surface is visible.

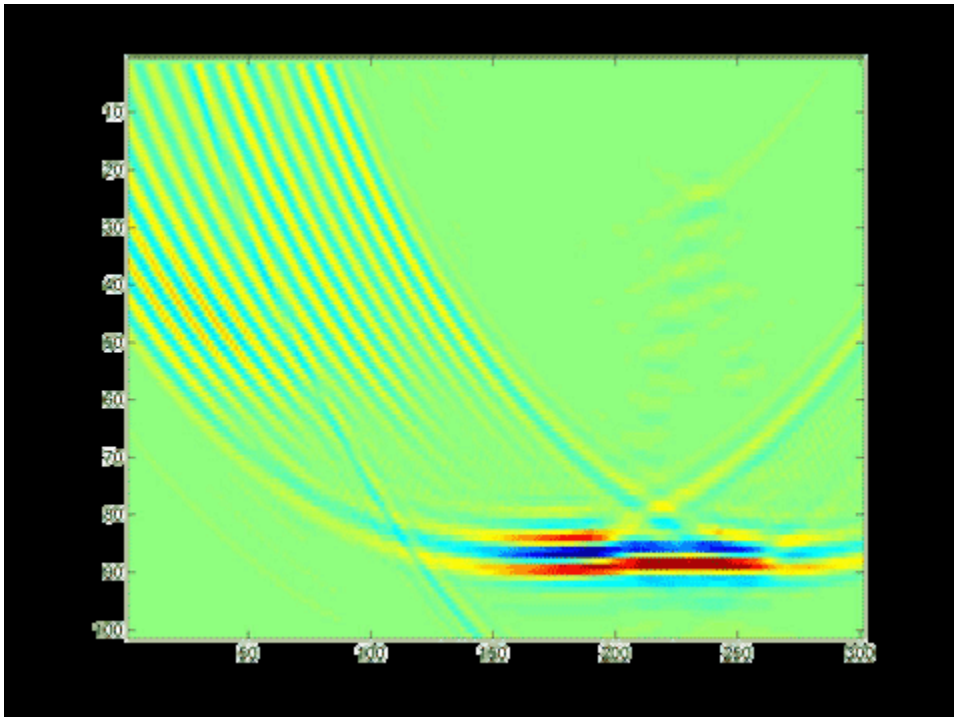
Below are images showing the summing of increasing numbers of ellipses. The reflecting surface is mostly horizontal, with a mountain barely visible on the right.



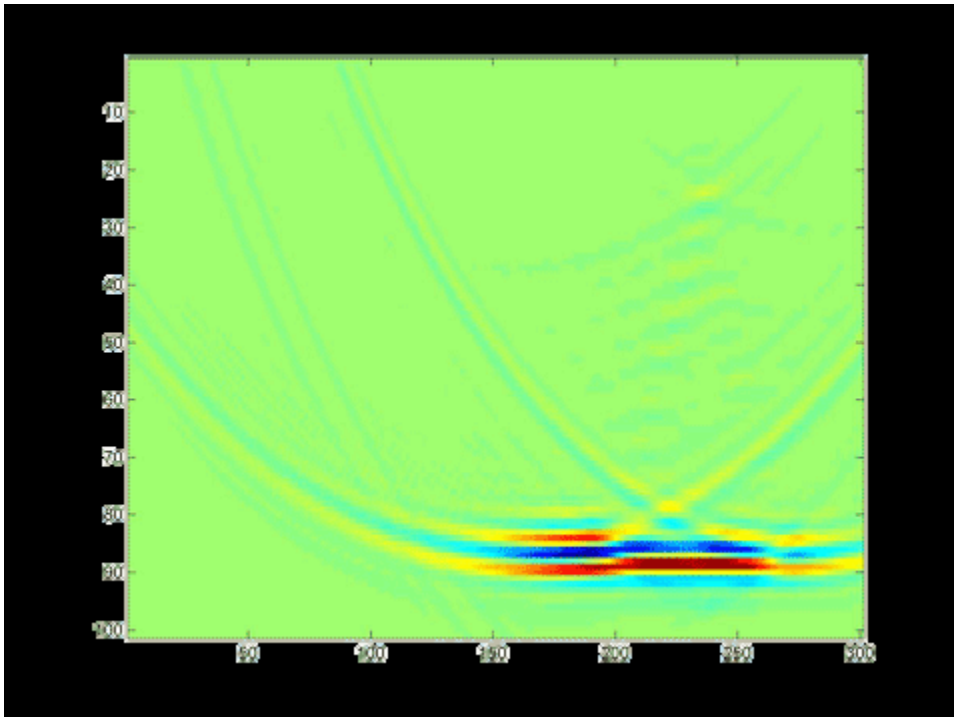
4 receivers



8 receivers



16 receivers



128 receivers

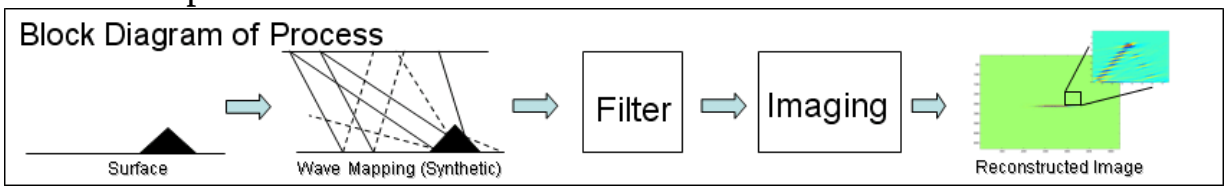
Overview of the Seismic Imaging Project

Having demonstrated how Kirchhoff Imaging techniques work, we may now outline and perform a brief demonstration of their implementations. In this section, we will give a general overview of our procedure and our model.

First we created bitmap images that represented surfaces to be imaged. We used two different images in this experiment, one of a mountain and one of the word "ELEC." We use these images to compare to the output of our imaging code.

We gave these files to Dr. Bill Symes who ran a simulated sesimic survey and returned time series vectors to us that represeneted the sound signals received.

Process Steps



Dr. Symes simulated the survey and then we processed the resulting time vectors

First we filtered our time vectors to eliminate any noise that may have been added in the survey. The image of the mountain was processed with no noise and the image of "ELEC" was processed with noise. When noise was present we analyzed the vectors in both the time and frequency domains to compare the effectiveness of our various algorithms

Finally we sent our filtered vectors to our imaging program that attempted to reconstruct the image by caluculating the ellipses of possible surface locations and adding them all up. The final results were analyzed to discuss the overall effectiveness of the procedure.

Introduction to Filtering

The data we get back from the seismic experiments will be in the form of a matrix of time series vectors. These vectors of course will have a certain amount of noise polluting the signal. In order to make any sense of the data we first have to filter these vectors to remove the unnecessary noise. It is only after filtering the data that we can hope to get a clear picture after imaging.

Ok, so let’s look at some simple models for a signal with noise. A simple case would be where the signal and the noise are in separate bandwidths in the frequency domain. With this knowledge we can chart a basic simple filtering algorithm.

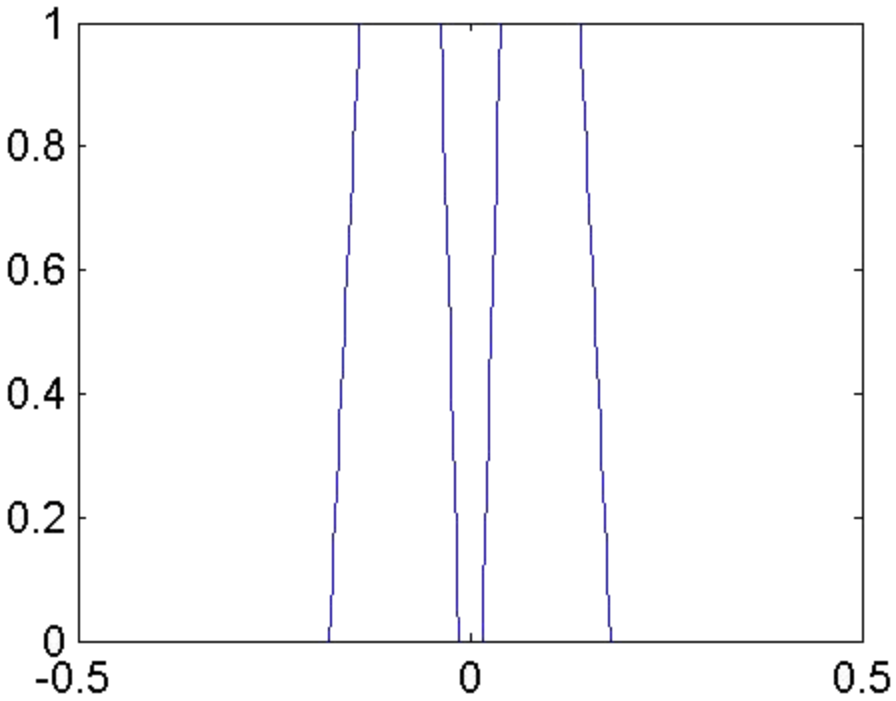
Step 1: Take the Fourier Transform of the data, so that we can look at it in the frequency domain.

Step 2: Looking at the spectrum we pinpoint the particular bandwidths where the signal is located and where the noise is located.

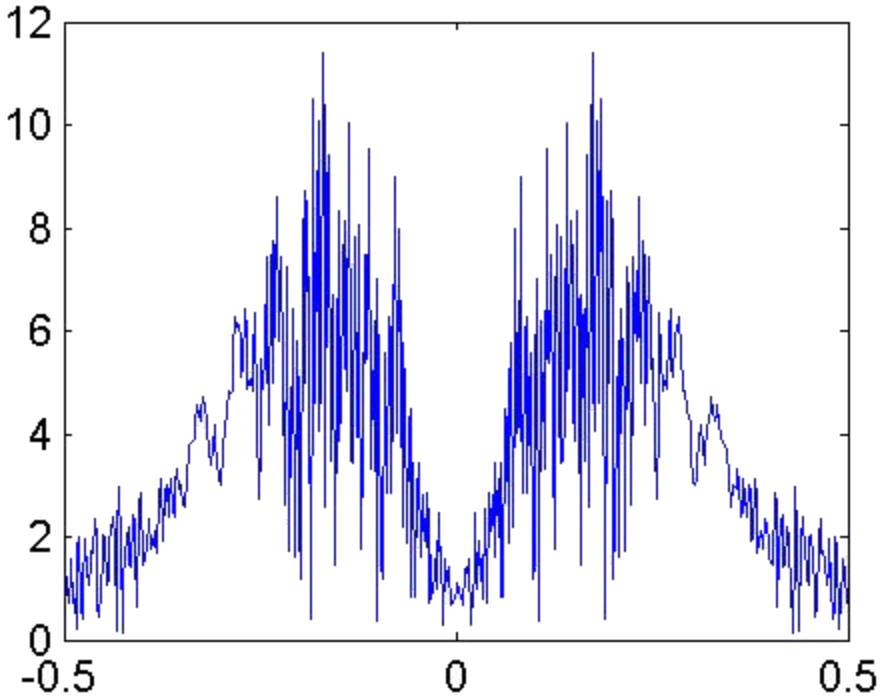
Step 3: We develop a standard Band pass filter that only allows the bandwidths in which the signal “lives” to come through.

Step 4: Now that we have our data outputted by the Band Pass filter, we are ready to send it for processing.

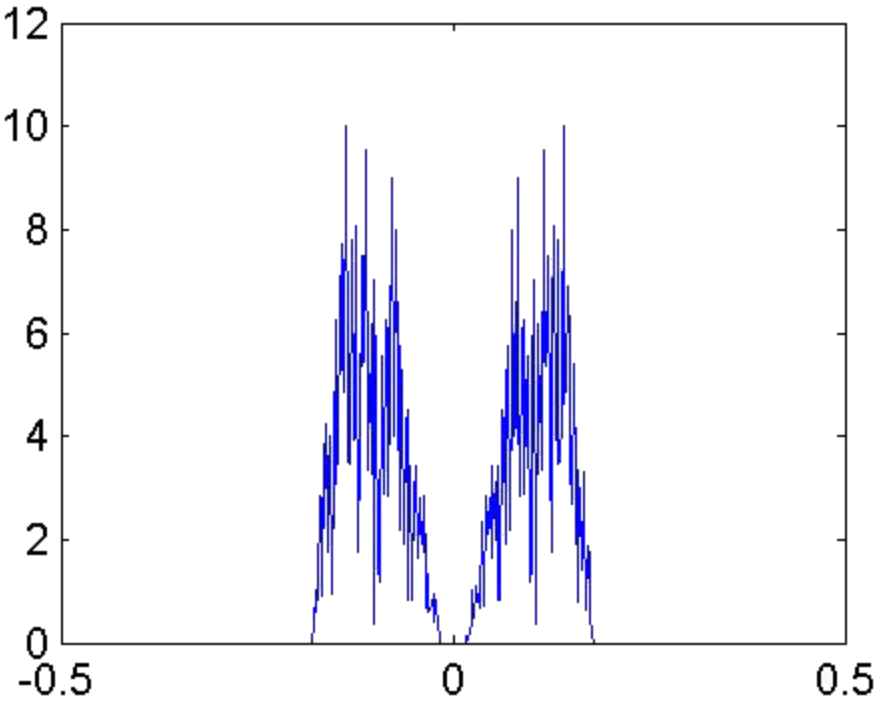
Bandpass filter



Unfiltered spectrum



Filtered Spectrum



This is a very simplistic model for filtering noise out of a signal. In real life, it is very unlikely that the signal and noise will live in separate bandwidths in the frequency domain. What is much more likely is that the signal and noise will overlap across a sizeable amount of the spectrum. Clearly no simple filter (like the Bandpass) can filter out the noise adequately from the received signal.

Clearly a more sophisticated approach for filtering is required. In the next section we will take a look at just such a technique; something so wonderful that has the potential to solve all of our problems.

The Wiener Filter

The wiener filter is an adaptive filter. It tailors itself to be the “best possible filter” for a given dataset. Below is a simplistic version of the derivation for the wiener formula.

Consider our standard equation to model a signal with noise:
Equation:

$$y[n] = x[n] + n[n]$$

We want to pass this y[n] through a filter ‘h’ such that we get back something that very closely resembles our original signal x, i.e. x .

So basically we want to design a filter that minimizes the difference between x and x . Lets start out by minimizing the least mean square error between x and x .
Equation:

$$\| x - \widetilde{x} \|_2$$

But we know x is h*y (h convolved with y), so we have:
Equation:

$$\| x - x * y \|_2$$

We can expand this expression known algebraic rules. Then we can take the Fourier transform of the expression to find the power spectra.
Equation:

$$\sum_j \left(X_j - H_j Y_j\right)^2$$

Equation:

$$\sum_j \left(X_j - H_j \left(X_j + N_j\right)\right)^2$$

We minimize this expression over H and in the end after all the simplification we get the following formula for H, our filter optimized to minimize the difference between x and x .

Equation:

$$H(f) = \frac{\left(|X(f)|\right)^2}{\left(|X(f)|\right)^2 + \left(|N(f)|\right)^2}$$

Where X(f) is the power of the signal and N(f) is the power of the noise.

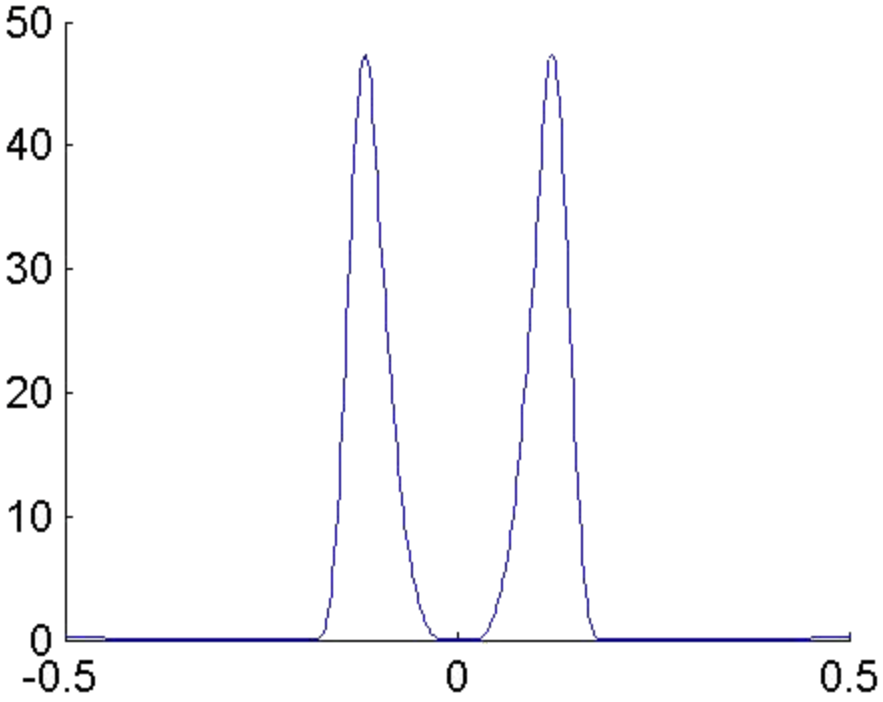
Just by inspection we can see that this filter will take care of the basics. For example when there is no noise, the filter response goes to just 1. When the signal is 0 the filter response goes to 0.

Notice to use this filter we will need to know the power spectrum of the actual signal. We also need to estimate the power spectrum of the noise. In real life this is done in numerous ways. Oftentimes out of convenience engineers will approximate the noise to Gaussian. This works well with varying results.

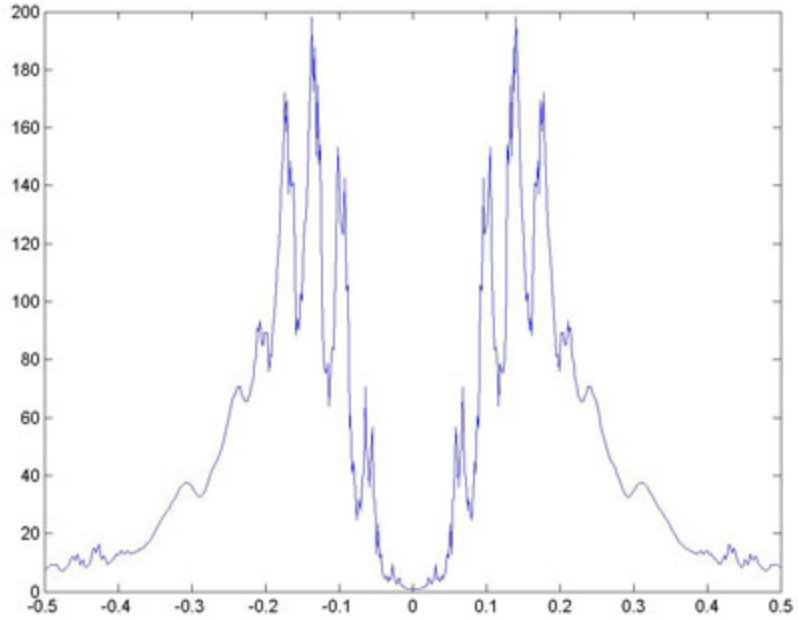
Building the Filter

One way to approximate the noise is “by inspection” or what is more commonly known as “guess and check”. In this method, you take a close look at your received signal and your pure signal.

Pure signal spectrum.



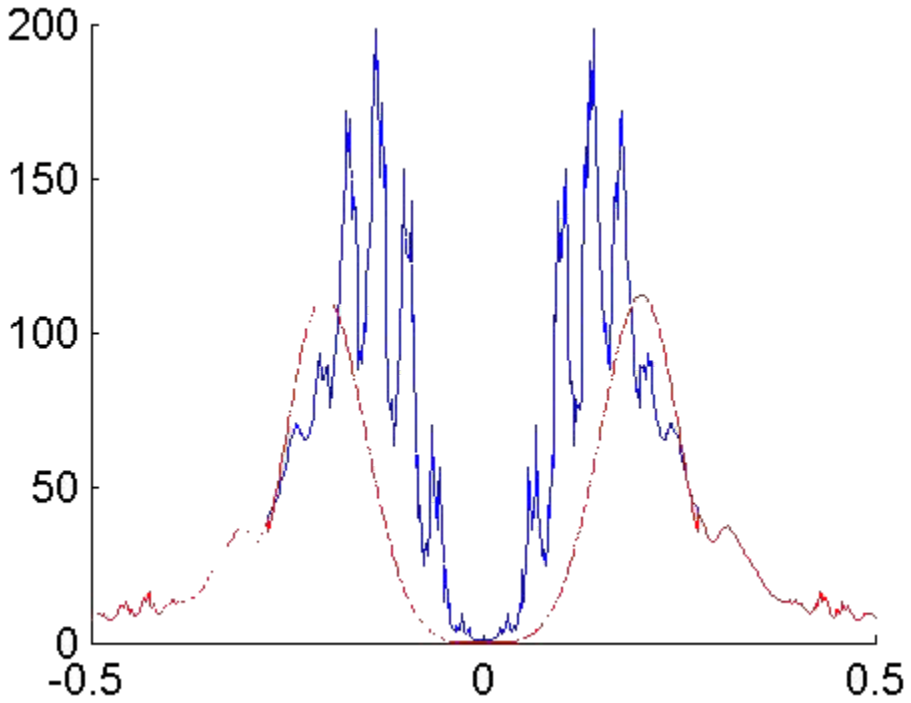
Received signal spectrum.



Then we try to figure out what the noise could look like based on this information. For this data set, we have traced out our “best guess” for the

noise spectrum below

Noise trace.



Just by looking at the noise spectrum we want, we look for patterns we could fit a mathematical formula to. In this example you can see that the function decays like a polynomial on one side and exponentially on the other.

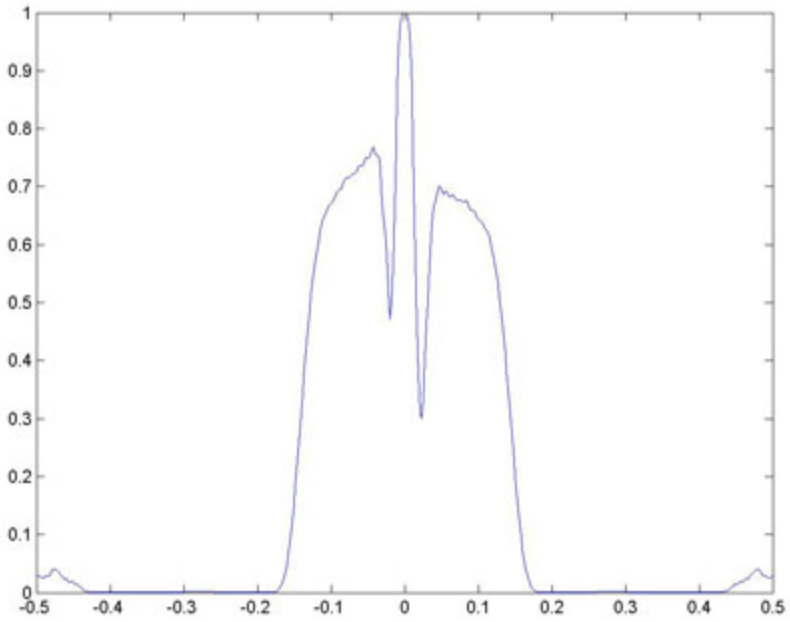
So our general noise toggling function becomes:
Equation:

And so we fix our parameters alpha, beta and gamma until we get something that resembles the function we drew in red earlier.

Note: This function is particular to this data set. For other types of noise, you would have to fit a new appropriate noise function.

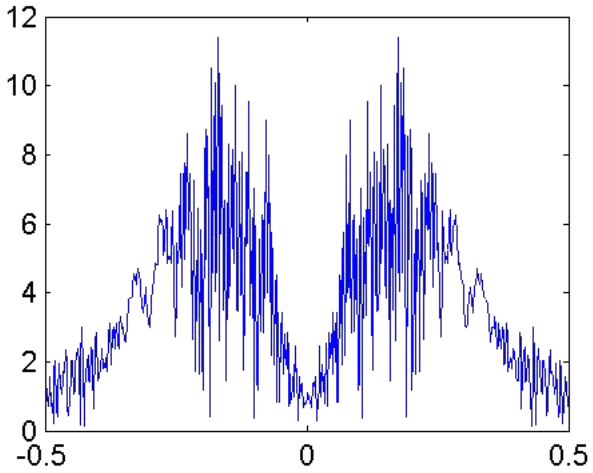
So now using the values we calculated for X(f) and N(f) we create our wiener filter.

Magnitude Plot of the Wiener filter.

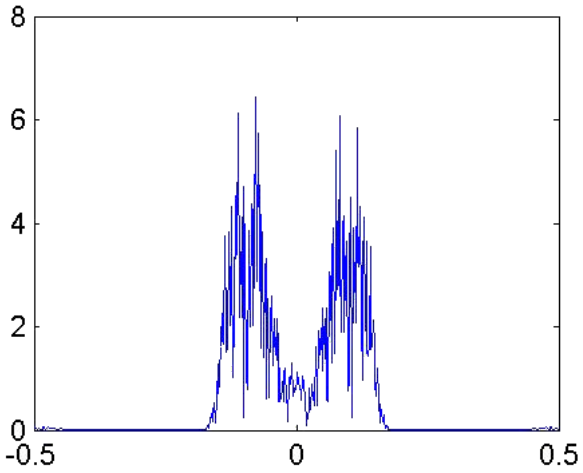


So lets try the filter on the data shall we?

Unfiltered data.



Filtereddata.



We see that the wiener filter does its job pretty well. It even wins over the bandpass filter in the simpler example since it even removes the excess amplitude added by the noise.

Lets take a look at the image of all the time series vectors.

A single unfiltered time series vector.

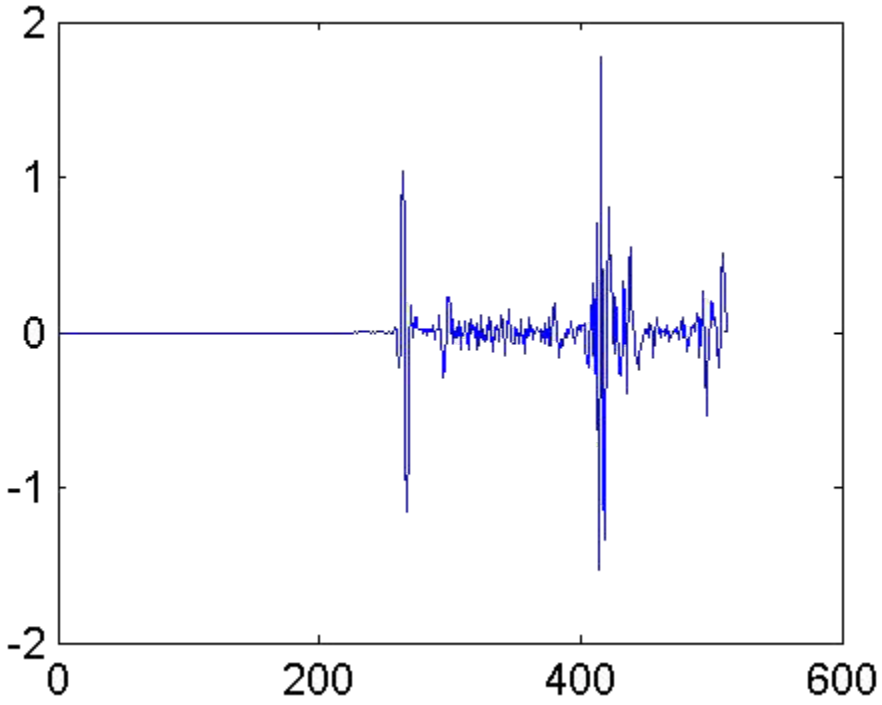
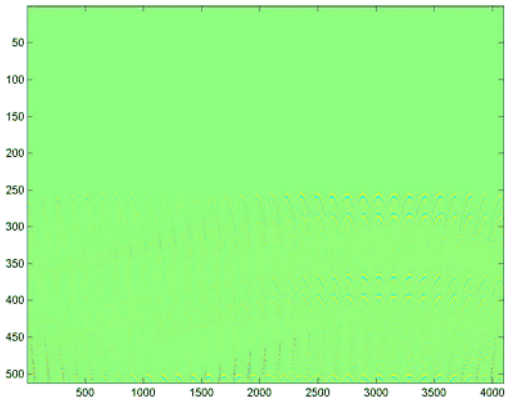
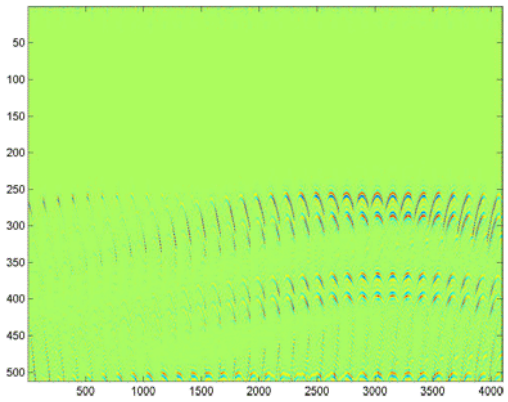


Image of unfiltered time vectors.

Image of filtered time vectors.





We can see that the unfiltered image is pretty blurry with not much distinguishable from the background. On the other hand the picture of the filtered version comes out pretty well and we can see distinct values for every time series.

So that concludes the section on filtering. Hopefully you have a better understanding on how to design adaptive filters, the wiener filter in particular. Now its time to move the data to the next stage of the process: Imaging.

Comparing Seismic Imaging Algorithms

The data from a seismic survey appears in the form of time series vectors, representing samples of the received acoustic signals over a fixed period of time. In our simulation example, the samples come every 4 ms (Fs = 250 Hz) and the vectors are 512 samples (2.048 s) in length. There is one time vector for each source/receiver combination, so for our example, we have 32 sources and 128 receivers for a total of 4096 time vectors. Therefore, our raw data will be represented as a 512x4096 matrix which we will load into MATLAB.

At this point, it is necessary to process this matrix and map this data onto a grid so that the shape of the formations we are studying may be determined. The basic technique involving ellipses has been described above, so all the program needs to do is to draw the ellipses, weighted by the magnitude of the time samples, and add the results for each time vector.

We consider now, two possible algorithms for drawing the necessary ellipses. The first method will traverse every pixel on the grid and at each point plot the value of the correct sample from the time vector. The second method will traverse the time vector and for each sample will plot an ellipse of appropriate magnitude.

What do we need to consider when comparing these two methods? First of all, there are 4096 vectors so this problem can become computationally very costly. For each method, we seek to find a standard run-time and evaluate different methods of optimizing this run time. Additionally, we hope to resolve as clear a picture as possible, so we should compare the graphs of the final answers to see which resolution is clearer.

Method 1 – Traversing the Grid

For every point, we can calculate the total distance required to travel from the source to that point and then to the receiver using the distance formula on the (X,Y) coordinates of the point, the source, and the receiver.
Equation:

$$D = \sqrt{(Y - Y_S)^2 + (X - X_S)^2} + \sqrt{(Y - Y_R)^2 + (X - X_R)^2}$$

We may now divide by velocity to get the time in seconds, and then we may again divide by the sample period to get an index for the time series vector that corresponds to this point.
Equation:

$$t_i = \frac{D}{VT_s}$$

Note that this value for time will not be an integer, so we must interpolate using the time series indices above and below it.
Equation:

$$t^+ = \text{ceil}\left(t_i\right)$$

Equation:

$$t^- = \text{floor}\left(t_i\right)$$

A simple linear interpolation method will give us an appropriate value that is weighted based on how close t is to t-minus and t-plus.
Equation:

$$\text{Mag} = \text{TimeSeries}\left(t^-\right)\left(t^+ - t_i\right) + \text{TimeSeries}\left(t^+\right)\left(t_i - t^-\right)$$

We finish by applying these equations to every point on the grid. This will trace out the ellipse patterns we desire for a given source-receiver pair.

Method 2 – Traverse the time vector

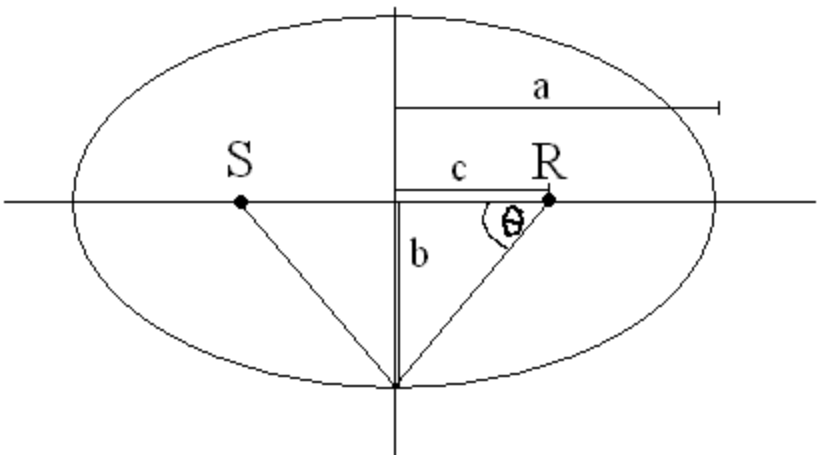
Taking one source-receiver pair's time vector, we first find the distance between the source and receiver. This distance is the minimum distance a signal must traverse. Since each sample in a time vector is 4ms, and we

know that a wave travels at 1500m/s, the **n**th sample in a time vector takes **n***4ms to travel **n***0.004s*1500m/s. If the **n**th sample distance less than the distance between the source and receiver, it is ignored since it is bad data. Usually these samples have a received signal value of 0 anyway.

If the sample qualifies as valid, we must find the ellipse which satisfies the condition that the sample distance equals the distance to the reflection surface and back. Using ellipse properties, we can see that the points of this ellipse can be found relative either the source or receiver.The equation for the ellipse in polar coordinates **(r,phi)** from one focus is:
Equation:

$$r = \frac{a\left(1 - e^2\right)}{e\cos(\mathrm{phi}) + 1}$$

where **e** is the eccentricity of the ellipse, **a** is the semi-major axes, and **c** is the distance of the focus to the center of the ellipse. The geometry for finding a and **c** is shown below:



Ellipse geometry.

We know **c** as the distance between the source and receiver divided by 2. **a** can be found by realizing that:
Equation:

$$b^2 = a^2 - c^2$$

b can be obtained since the hypotenuse of each right triangle is equal to half the sample distance. A little trigonometry takes care of finding **b**:
Equation:

$$\text{theta} = \text{acos}\left(\frac{c}{\text{hypotenuse}}\right)$$

Equation:

$$b = \text{hypotenuse} * \sin(\text{theta})$$

Eccentricity **e** is simply defined as:
Equation:

$$e = \frac{c}{a}$$

We then plug all this into **Equation (6)** and get a vector of radii for all **phi** between pi and 2*pi (the negative half of the unit circle).

This vector is mapped to the image grid via the matlab command **pol2cart** and the shifted by **c** to bring the ellipse into line with the focii (source and receiver).

Finally, the value of the time sample is then added to these grid coordinates. The process is repeated for each source-receiver time vector.

Time Comparison

Both methods require treating each of the 4096 vectors separately and traversing the vectors one by one. Therefore, we can talk about efficiency in terms of the time required to process one vector and extrapolate this to total processing time. Without any optimizations, both methods take several seconds to process one vector, which means that total process time is on the order of 5-10 hours. So how can we reduce these times? Well, one obvious answer would be to use a different computing simulator or work with a language more optimized for this type of processing. However, for our purposes, let's only consider optimizations to the actual algorithms.

For method one, the main factor we can control is the size of the grid. We may greatly reduce our processing time by simply searching a smaller grid. The idea is to traverse the entire grid for the first few vectors and then to only traverse the areas where the magnitude is greater than some threshold. It is possible, in this way, to decrease the area by a factor of 2-10 and thus improve the speed.

For method two, we set a threshold value and only consider time samples above this value. This is helpful in eliminating all of the zero values in the early part of the time vectors that occur before any pulse returns. We must be careful however in this optimization because if we set the threshold too high, we will be hurting our resolution and/or creating noise in the graph.

	Algorithm	
Experiment	**Method 1**	**Method 2**
Mountain (Full Algorithm)	10-12 sec	5 sec
Mountain (Optimized)	3-4 sec	1-2 sec
ELEC (Optimized)	5-6 sec	3-4 sec

Algorithm Timing Comparison (approx. times)

Resolution Comparison

Here are two pictures of the same image reconstructed using each method

Image from Method 1

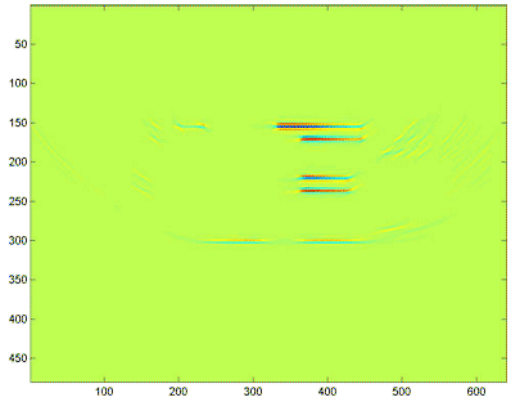
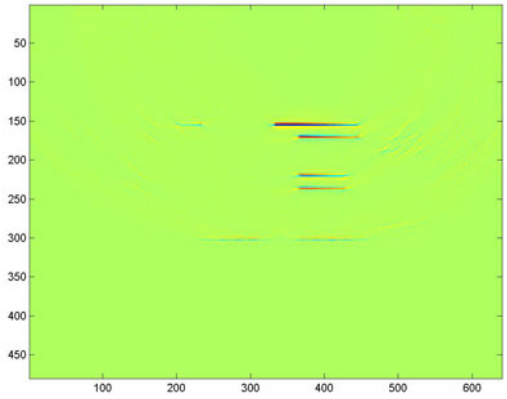


Image from Method 2



As you can see, Method 1 shows details much more clearly because it uses a linear interpolation formula, while Method 2 rounds off the ellipse coordinates to fit them to the grid. This means that method one should be more accurate and should show less error from the discrete nature of the samples. Method 2 is faster, but method 1 shows higher resolution.

You can download copies of the matlab code for method 1, the optimizing wrapper for method 1 and method 2.

Seismic Imaging Project Results

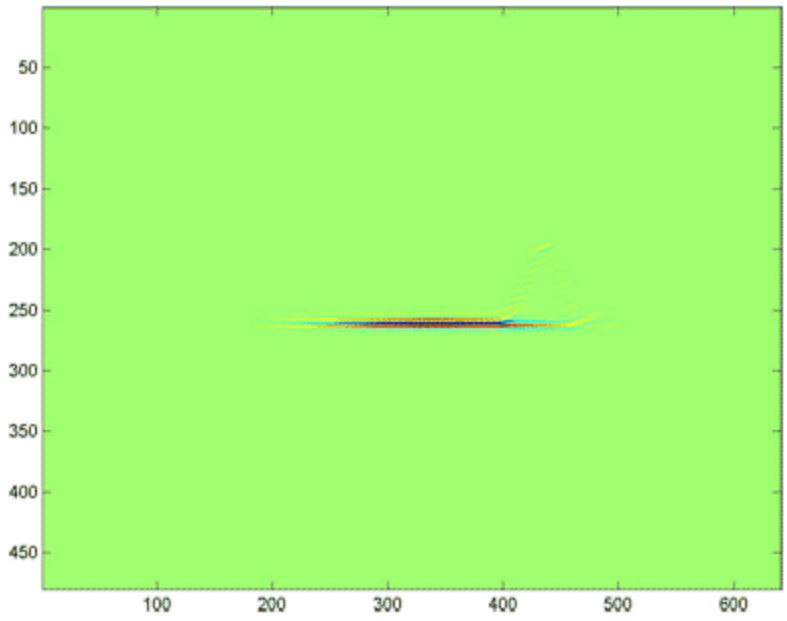
Now we come to the results of our work

First we fed this picture into our imaging system as a simple example:

A surface with a mountain.

This system had no added noise, i.e. we are imaging the pure signal. This is what we got:

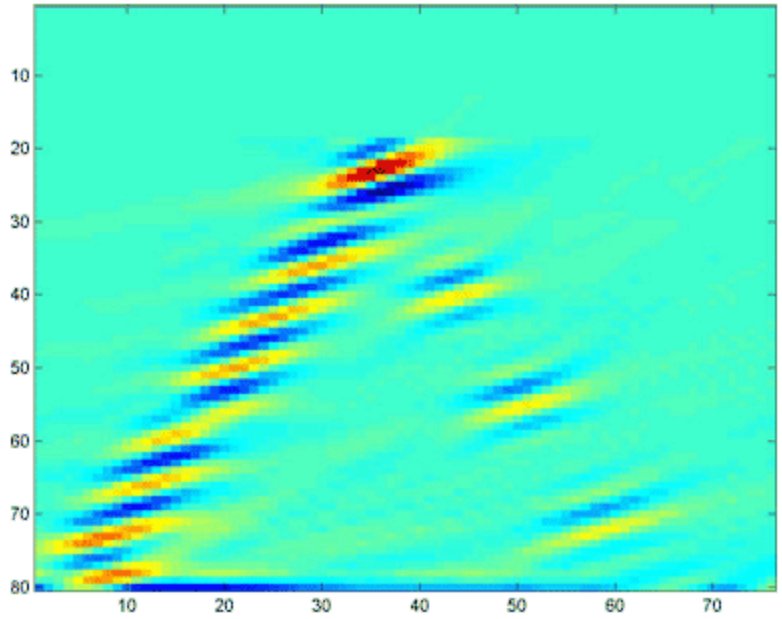
The surface with a mountain after it has been run through the entire process.



We see the horizontal plane is very clearly resolved. This is because most of the incident power is received by the receivers. The mountain is a little faint compared to the horizontal plane. This is because of the slanting nature of the surface: i.e. not all of the power is reflected towards the receiver; a sizeable amount of it is reflected off at odd angles and never reaches the receivers.

Let’s take a look at the blowup of the mountain itself. We see that the mountain is still clearly resolved against the background. We see that the slope is stepped: this is because we image the edges of each pixel at a time. The side of the mountain facing the sources is imaged still clearer than the lee side since very few source waves manage to reflect off the lee side.

A close-up of the mountain.



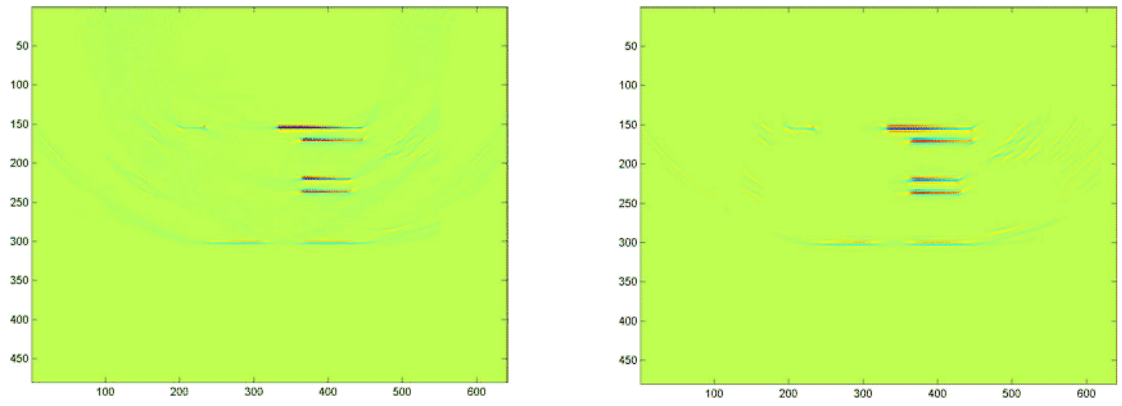
Lets take a look at another more complex image:

ELEC

This picture data came with “juicy” noise. Below we have imaged both the filtered and unfiltered versions of this data set.

ELEC imaged without filtering of the raw data.

ELEC imaged with filtering of the raw data.



ELEC images.

We notice in both we can make out the horizontal surface portions of the middle E and L. The horizontal portions are pretty much lost. The first E not really visible and the top curve of the C is faintly visible. The filtered portion does have better resolution than its unfiltered counterpart: The outer sides are smudged in the unfiltered one and the C in particular is more visible in the filtered version. The horizontal portions in the center E and L return so much of the signal that the noise is overwhelmed for the most part. We understand that filtering is most visible in the detailed parts of the picture which is why the horizontal surfaces are clearly resolved in both the filtered and unfiltered whereas the outer regions of the picture have smudges in the unfiltered version that vanish in the filtered version.

So what have we learned about our imaging process:

"1. Horizontal surfaces are clearly visible since they return so much of the power sent to them straight back to the receivers."

"2. Positions of the sources and receivers matter. Had not the first E been out of source-receiver range, we could have gotten more clarity."

"3. Vertical surfaces are exceedingly difficult to image, given the positions of sources and receivers we are using."

"4. Good resolution at high elevations. Algorithm needs to be modified if it has to deal with multiple layers of surfaces."

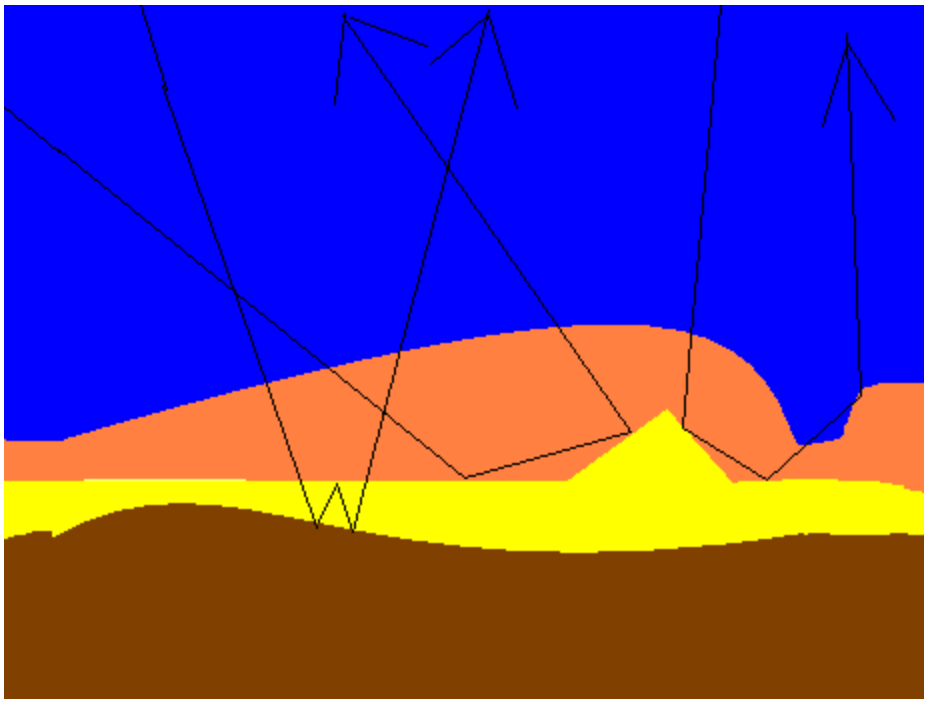
Possible Extensions to the Seismic Imaging Project

We hope that we have succeeded in covering the basics of seismic image reconstruction, but we know that there are many more problems to be explored in this area. For those readers interested in learning more about these topics, here are some ideas for extensions that we hope will be helpful to your endeavors.

We showed many different aspects of this technique through our simulations, but there are still many factors in our simulation model that could be varied to gain greater insight. For instance, we used the same positioning of sources and receivers in both tests. From the imaging of “ELEC”, we saw that there were forms at the edges of the map that we could not resolve clearly. A good and simple modification to our experiment would be to compare images of the same map, using different source and receiver positions. Another good experiment would be to use surfaces with greater detail. It would be interesting to see how these surfaces would resolve and it would also provide a better test for our filtering algorithms to see how they affect detailed images.

Additionally, our Imaging code assumes that there are two uniform media, one of which reflects the pulse entirely and one of which transmits it entirely. We did not account for layers of different types or for multiple reflections within the surfaces themselves. Either of these considerations would increase the complexity of the imaging code but would give it greater flexibility for resolving different types of surfaces. Creating a test image for this would be fairly easy.

Multiple Reflection Layers



One possible extension is to consider the possibility of multiple reflections.

Another factor that our algorithm depends on is using a known speed of sound in the medium we are imaging. In practice this is a precarious assumption to make. However, the speed can be calculated fairly reliably by a test using one pulse and two different receivers. Comparing data from these two receivers would allow you to calculate the velocity. This is a dimension of realism that could be added fairly easily.

Our estimation of the noise power was fairly crude, and we felt that the Wiener filtering could have easily been made stronger by either studying more detailed noise profiles or using a statistical method to fit the estimator function. Alternatively, we might use Fourier analysis to isolate the noise spectrum from the spectrum of the signal plus noise.

Finally, we have discussed one very straight forward way of interpreting the data: filtering the vectors one by one and plotting the corresponding ellipses and summing the results. However, there should be other ways of analyzing

this data in order to see shapes and pictures. Learning about Deconvolution or Radon Transform algorithms and comparing resolution and filtering techniques would be a great new dimension to add to a future project.

Acknowledgements

http://www.trip.caam.rice.edu/ "Introduction to the Mathematics of Seismic Inverse Scattering," CAAM Graduate Research Seminar, September 1, 2004. (Photograph of surface ship firing waves at ocean floor and diagram of ellipses taken from this source)

http://www.trip.caam.rice.edu/ "Short Course on Mathematical Foundations of Reflection Seismology,“ Summer 2004.

http://oldsite.vislab.usyd.edu.au/ Rosalind Wang, "Wiener Filtering.“

And most of all, an enormous thanks to Dr. Bill Symes for simulating the reflection data and guiding us in our work.